Computing Linear Regions in Neural Networks with Skip Connections

Jan Verschelde[†]
joint with Johnny Joyce

University of Illinois at Chicago
Department of Mathematics, Statistics, and Computer Science
`http://www.math.uic.edu/~jan`
`janv@uic.edu`

The 27th Workshop on Computer Algebra in Scientific Computing
24-28 November 2025, Dubai, United Arab Emirates

[†]Supported by a 2023 Simons Travel Award.

Outline

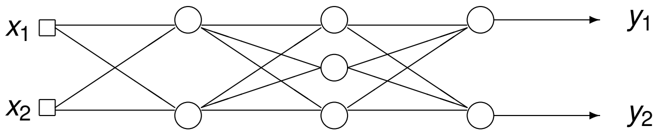
Computing Linear Regions in Neural Networks

1. Introduction
   - neural networks with skip connections
   - tropical geometry and linear regions

2. Computing Linear Regions
   - recurrences and algorithms
   - a small example network

3. Experiments and Applications
   - visualization of linear regions
   - comparing networks with and without skip connections
   - caching linear regions after training

an example of a neural network



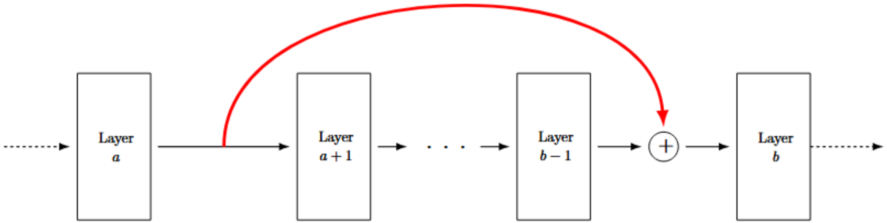
represents a neural network with two inputs x_1, x_2, two outputs y_1, y_2, and three layers

$$\begin{aligned} z_1 &= \nu\left(A^{(1)}x + b_1\right) \\ z_2 &= \nu\left(A^{(2)}z_1 + b_2\right) \\ y &= \nu\left(A^{(3)}z_2 + b_3\right) \end{aligned}$$

where $A^{(1)}, A^{(2)}, A^{(3)}$ are the weights, b_1, b_2, b_3, are the biases, and ν is an activation function.

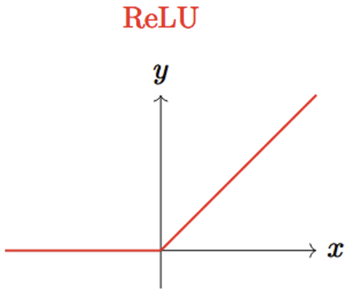
skip connections

Skip connections (or shortcut connections, or residual connections) add the output of a particular layer to the input of a later layer.



`ResNet` [He, Zhang, Ren, Sun, IEEE, 2016]
uses deep residual learning for image recognition.

a piecewise linear activation function



The output of the first layer of a neural network is given by:

$$\nu(x) = \texttt{ReLU}(Ax + b) = \max(Ax + b, 0),$$

where A is the weight matrix of the first layer,
and b is the corresponding bias.

As the `ReLU` activation function is piecewise linear, the nonlinearity of the neural network can be studied via tropical geometry.

tropical algebra

Decompose the matrix A as $A := A_+ - A_-$

- where the (i,j)-th entry of A_+ is $\max\{a_{ij}, 0\}$, and
- where the (i,j)-th entry of A_- is $\max\{-a_{ij}, 0\}$.

Then layer $\nu(x)$ is the difference of two tropical polynomials:

$$\begin{aligned} \nu(x) &= \max\{A_+ x + b, A_- x\} - A_- x \\ &= \left((x^{\otimes A_+} \otimes b) \oplus x^{\otimes A_-} \right) \oslash A_- x \end{aligned}$$

where

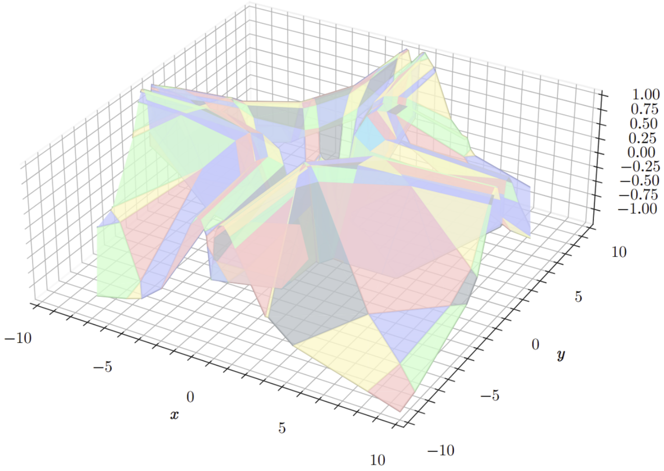
$$x \oplus y = \max\{x, y\}, \quad x \otimes y = x + y, \quad x \oslash y = x + (-y) = x - y,$$

and

$$x^{\odot n} = x \cdot n.$$

linear regions

A linear region in a neural network is a connected region for which the map defined by the neural network is linear.



literature

- R. Balestriero and R. G. Baraniuk.
  Mad Max: Affine spline insights into deep learning.
  Proceedings of the IEEE, 109(5):704–727, 2021.
- R. Balestriero, A. I. Humayun, and R. G. Baraniuk.
  On the geometry of deep learning. *Notices of the American Mathematical Society*, 72(4):374–385, 2025.
- L. Zhang, G. Naitzat, and L.-H. Lim. **Tropical geometry of deep neural networks.** In *International Conference on Machine Learning*, pages 5824–5832. PMLR, 2018.
- J. Joyce. *Algebraic Frameworks and Computational Experiments for Enhanced Machine Learning*. PhD thesis, UIC, 2025.

`https://github.com/johnnyvjoyce/tropical-geometry-linear-regions`

Computing Linear Regions in Neural Networks

a recurrence relation gives the result

Let f be an L-layered neural network with ReLU activations and with n_1, n_2, $\ldots$, n_L neurons on each layer.

Then any set of vectors $\{v_\ell \in \{-1, 1\}^{n_\ell} \mid \ell \in [L]\}$ uniquely specifies a single linear region of f, where the region is given by:

$$\{x \in \mathbb{R}^n \mid \widetilde{H}(x, \ell) \leq \widetilde{G}(x, \ell) \text{ for all } \ell \in [L]\}$$

where

$$\begin{aligned} \widetilde{H}(x, \ell) &:= \texttt{diag}(v_\ell) \left(H_A^{(\ell)}(x)x + h_b^{(\ell)}(x) \right) \\ \widetilde{G}(x, \ell) &:= \texttt{diag}(v_\ell) \left(G_A^{(\ell)}(x)x + g_b^{(\ell)}(x) \right) \end{aligned}$$

with H and G as in the summands in the tropical polynomial of f,

and where $\texttt{diag}(v_\ell) = \displaystyle\sum_{i=1}^{n_\ell} \mathbf{e}_i^T v_\ell \mathbf{e}_i$, with $\mathbf{e}_i$ the i-th unit vector.

method to find linear regions

```
1: function FindLinearRegion($v_1, \ldots, v_L$)
2:     $F_A^{(0)}, G_A^{(0)}, H_A^{(0)}, f_b^{(0)}, g_b^{(0)}, h_b^{(0)} \leftarrow$ Initial values laid out in (4)
3:     $S \leftarrow$ Empty array // To be filled with inequalities that define the linear
       region.
4:     for $l = 0, \ldots, L-1$ do
5:         $A \leftarrow$ weight matrix of layer $\ell + 1$
6:         $b \leftarrow$ bias vector of layer $\ell + 1$
7:         $G_A^{(l+1)}, H_A^{(l+1)}, g_b^{(\ell+1)}, h_b^{(\ell+1)} \leftarrow$ Values for the next layer laid out in (3)
8:         $F_A^{(\ell+1)} \leftarrow$ a blank matrix to be filled out
9:         $f_b^{(\ell+1)} \leftarrow$ a blank vector to be filled out
10:        for $i = 1, \ldots, l$ do
11:            $\vec{\alpha} \leftarrow$ $i$-th row of $H_A^{(\ell+1)} - G_A^{(\ell+1)}$
12:            $\beta \leftarrow$ $i$-th entry of $h_b^{(\ell+1)} - g_b^{(\ell+1)}$
13:            if $(v_{l+1})_i == +1$ then
14:                Append the inequality $\vec{\alpha} \cdot x + \beta \leq 0$ to $S$
15:                $i$-th row of $F_A^{(\ell+1)} \leftarrow$ $i$-th row of $G_A^{(\ell+1)}$
16:                $i$-th entry of $f_b^{(\ell+1)} \leftarrow$ $i$-th entry of $g_b^{(\ell+1)}$
17:            else
18:                Append the inequality $-\vec{\alpha} \cdot x - \beta \leq 0$ to $S$
19:                $i$-th row of $F_A^{(\ell+1)} \leftarrow$ $i$-th row of $H_A^{(\ell+1)}$
20:                $i$-th entry of $f_b^{(\ell+1)} \leftarrow$ $i$-th entry of $h_b^{(\ell+1)}$
21:        if the region given by the intersection of elements of $S$ is empty then
22:            return Message indicating that the region is empty,
               and specify the current index $\ell$ where we reached an empty set
23:    return S
```

tree traversal to find linear regions

```
1: results ← a global array, accessible within nested calls
2: function TRAVERSE(depth, $v$)
   // $v$ is some set of elements $\{v^{(1)}, \ldots, v^{(k)}\}$. Initially, $v$ is empty.
3:     if depth == $L$ then
4:         msg ← FINDLINEARREGION($v$) // Call algorithm 1.
5:         if msg shows an empty region then
6:             return index of failure
7:         else
8:             Append msg containing linear region to global array of results
9:             return nothing
10:    else
11:        for each possible assignment of $-1$ and $+1$ to entries of $v_{\text{depth}}$ do
12:            $v'$ ← current assignment of $-1$ and $+1$
13:            msg ← TRAVERSE(depth+1, $v \cup v'$) // Recursive call.
14:            if msg contains index of a failure, and current depth > index of failure
               then
15:                return index of failure
```

complexity

The number of linear regions is exponential in the number of layers.

The tree traversal has the benefit over the brute-force method that it can abandon large branches of the tree if empty at an early layer as is the case for truth assignments outside the training data.

However, the tree has as many leaves as $2^{\#\text{neurons}}$.

a small example network

Consider a network

- with a 2-dimensional input layer,
- 2 hidden layers with 2 neurons per layer,
- and a 1-dimensional output layer.

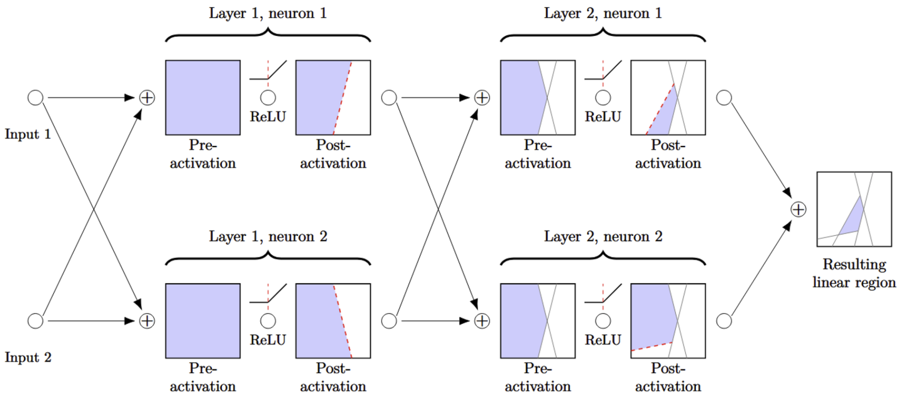
The weights and biases of each hidden layer in the network are:

$$A^{(1)} = \begin{bmatrix} -4 & 1 \\ -4 & -1 \end{bmatrix}, \quad b^{(1)} = \begin{bmatrix} 2 \\ 3 \end{bmatrix},$$

$$A^{(2)} = \begin{bmatrix} -8 & 3 \\ -\frac{21}{4} & \frac{19}{4} \end{bmatrix}, \quad b^{(2)} = \begin{bmatrix} -4 \\ 1 \end{bmatrix}.$$

We assume that the 1-dimensional output layer does not have an activation function, so we can focus on the two hidden layers.

walkthrough on the small example network



the search tree of the small example network

$$\begin{tikzcd}[column sep=tiny]
 & & & & & & & v_1, v_2 \arrow[dllllll] \arrow[dll] \arrow[drr] \arrow[drrrrrr, thick] & & & & & & & & \\
 & \left({-1 \atop -1}\right), v_2 \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & & & \left({-1 \atop +1}\right), v_2 \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & & & \left({+1 \atop -1}\right), v_2 \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & & & \left({+1 \atop +1}\right), v_2 \arrow[dl] \arrow[d] \arrow[dr, thick] \arrow[drr] & & \\
\left({-1 \atop -1}\right),\left({-1 \atop -1}\right) & \left({-1 \atop -1}\right),\left({-1 \atop -1}\right) & \left({-1 \atop -1}\right),\left({+1 \atop -1}\right) & \left({-1 \atop -1}\right),\left({+1 \atop +1}\right) & \left({-1 \atop +1}\right),\left({-1 \atop -1}\right) & \left({-1 \atop +1}\right),\left({-1 \atop -1}\right) & \left({-1 \atop +1}\right),\left({+1 \atop -1}\right) & \left({-1 \atop +1}\right),\left({+1 \atop +1}\right) & \left({+1 \atop -1}\right),\left({-1 \atop -1}\right) & \left({+1 \atop -1}\right),\left({-1 \atop -1}\right) & \left({+1 \atop -1}\right),\left({+1 \atop -1}\right) & \left({+1 \atop -1}\right),\left({+1 \atop +1}\right) & \left({+1 \atop +1}\right),\left({-1 \atop -1}\right) & \left({+1 \atop +1}\right),\left({-1 \atop -1}\right) & \left({+1 \atop +1}\right),\left({+1 \atop -1}\right) & \left({+1 \atop +1}\right),\left({+1 \atop +1}\right)
\end{tikzcd}$$

Computing Linear Regions in Neural Networks

1. Introduction
   - neural networks with skip connections
   - tropical geometry and linear regions
2. Computing Linear Regions
   - recurrences and algorithms
   - a small example network
3. Experiments and Applications
   - visualization of linear regions
   - comparing networks with and without skip connections
   - caching linear regions after training

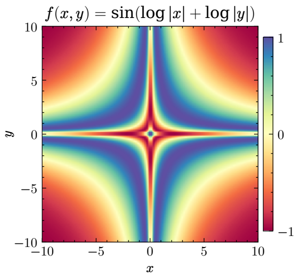
regression problems

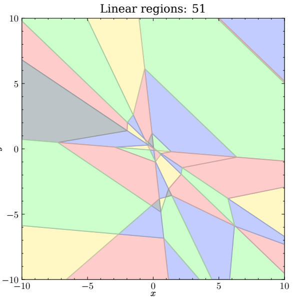
Each function has two scalar input variables x and y, and one scalar output:

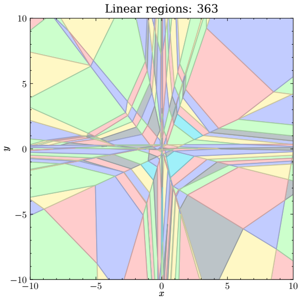
$$\begin{aligned}
\phi_1(x,y) &= \sin(\log|x| + \log|y|) \\
\phi_2(x,y) &= \sin\left(\sqrt{x^2+y^2}\right) \div \sqrt{x^2+y^2} \\
\phi_3(x,y) &= \sin\big(\cos(x/2)\big)\sin\big(\cos(y/2)\big) \\
\phi_4(x,y) &= \sin\big(\tan(x/2)\big)\sin\big(\tan(y/2)\big)
\end{aligned}$$

We expect ϕ_1, ϕ_2, ϕ_3 to be modeled well by neural networks, whereas overfitting is expected on ϕ_4.

using two and five layers on ϕ_1

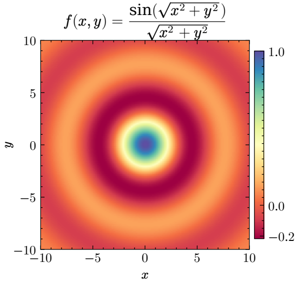


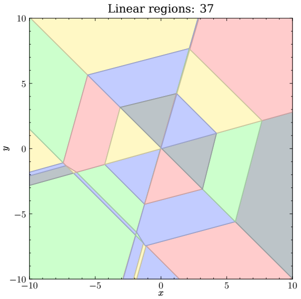


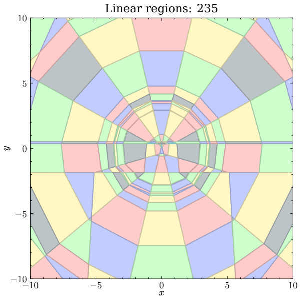


- The 5-layer network partitioned the input into regions that mimic the patterns in the training data, producing a star-like pattern.
- The 2-layer network also produced visual patterns matching the training data in less detail.

using two and five layers on ϕ_2

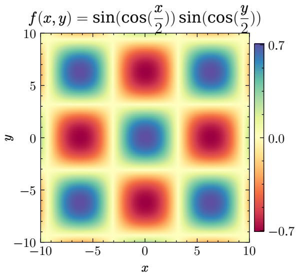


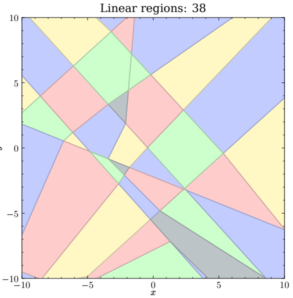


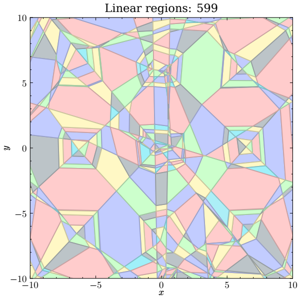


- The 5-layer network partitioned the input into regions that mimic the patterns in the training data, producing a radial pattern.
- The 2-layer network also produced visual patterns matching the training data in less detail.

using two and five layers on ϕ_3

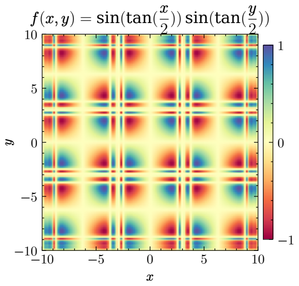


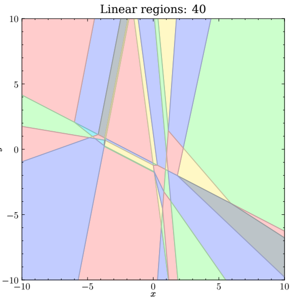


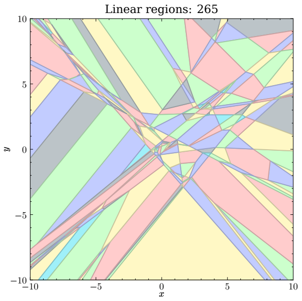


Despite contours more difficult than ϕ_1 and ϕ_2,
the patterns produced by the 5-layer network are still sensible.

using two and five layers on ϕ_4



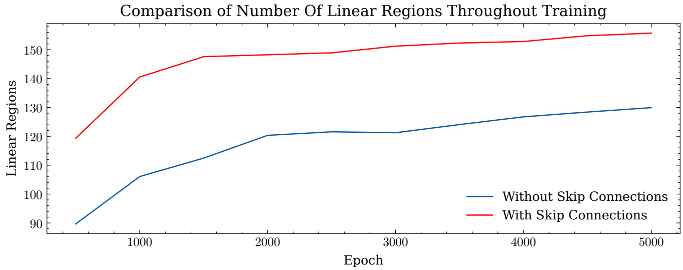


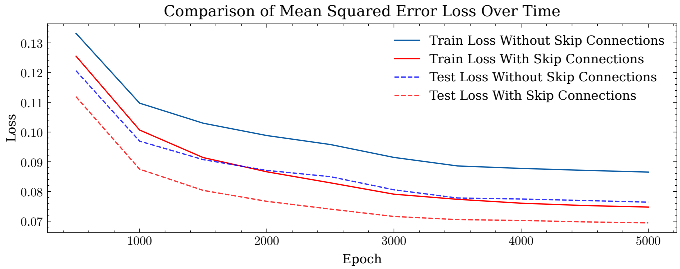


Overfitting was expected and the network produced clusters of tiny linear regions near the center.

According to intuive understanding, overfitting occurs when too much emphasis is placed on noise in training data.

comparing networks with and without skip connections





caching values in neural networks

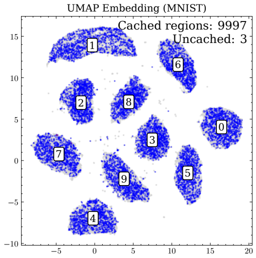
What can we do with the linear regions?

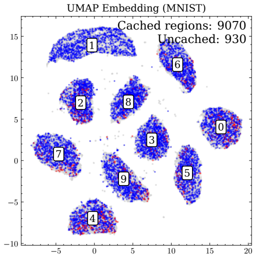
We propose a method to compress nonlinear neural networks into one single layer.

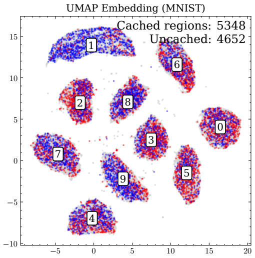
Experiments are on classifier models on three different data sets:

1. MNIST data set of handwritten digits,
2. CIFAR-10 data set of images,
3. Street View House Number (SVHN) data set.

rates of caching for 8, 16, 32 neurons on MNIST

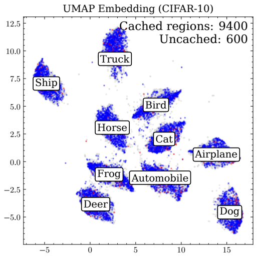


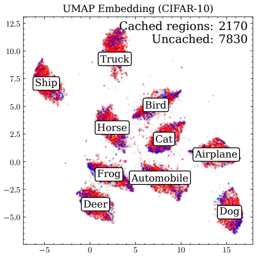


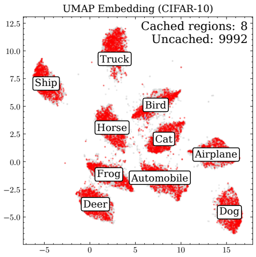


- Blue points denote test data in the same linear region as at least 1 training set instance.
- Red points are *not* in the same region as any training instance.
- Gray points represent training data.

rates of caching for 8, 16, 32 neurons on CIFAR-10

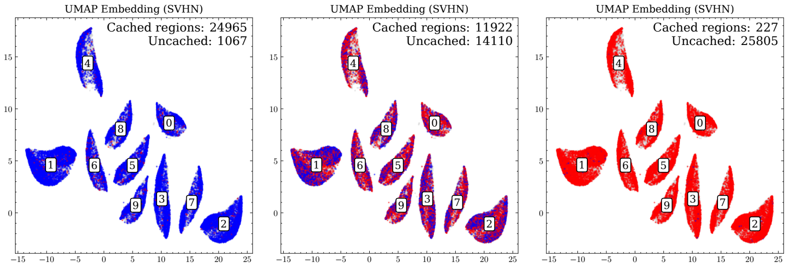






- Blue points denote test data in the same linear region as at least 1 training set instance.
- Red points are *not* in the same region as any training instance.
- Gray points represent training data.

rates of caching for 8, 16, 32 neurons on SVHN



- Blue points denote test data in the same linear region as at least 1 training set instance.
- Red points are *not* in the same region as any training instance.
- Gray points represent training data.

conclusions

- Recurrence relations form the basis for algorithms to compute all linear regions of a neural network, generalized to work for networks with skip connections.
- Visualizations of those linear regions help to understand how patterns in the training data may lead to overfitting.
- Caching the linear transformations allows for faster predictions.
- Through experimentation, we found that skip connections allow a model to both be more expressive by creating more advanced output maps, while also ensuring that these maps can generalize to new data.